1.Посчитать по получившимся данным в eax начальное данное в переменной и мусор (то есть не забыть выход за пределы переменной при вводе в случае его наличия).
2.Реализовать на ассемблере фрагмент си-кода. Учесть побочные эффекты, значение выражения поместить в eax. Учесть короткую логику. *p – значения указателя созданного, как, например, int *p, в ассемблере требует разыменования, p – само значения адреса, dword[p]. При test eax, eax помнить, что учитывается edx, то есть обнулять его заранее.
3.Для доступа к элементу двумерного массива в ассемблере index1 умножается на количество столбцов и к нему прибавляется номер столбца, index2, где index1 – номер строки, index2 – номер столбца.
4.Написать switch для меток ассемблерного кода. Количество случаев, надо запомнить, всегда ограничено, ifов скорее всего не должно быть. Не забывать break. Если два числа на один case, то нельзя писать
case 1, 3:

_____
break;
но можно писать
case 1:
case 3:

_____
break; тогда в case 1 также выполнится case 3.
Не забывать default. Как правило, метки распределены по каким-то реальным фактам.
5.ds – это канарейка. Канарейка ставится в стек для фиксации изменения в её значении в случае уезда по стеку. Помнить про cdecl/stdcall/fastcall. Помнить про использование epb как frame pointer (указатель фрейма). Нарисовать стек верно, не забыть про размещение локальных переменных на стеке и в случае стандарта stdcall возвращаемого элемента над адресом возврата. Фиксировать все изменения esp (особенно в случае отказа от ebp, указателя фрейма).
6.Перевод чисел в двоичный формат IEEE 754. Алгоритм: перевести целую часть числа в двоичную систему, перевести дробную часть числа в двоичную систему, записать. Сдвинуться так, чтобы запятая стояла после единственной единицы: 0001, … Пусть сдвинулись на n позиций. Пусть e – число бит, которыми кодируется поле экспоненты. Поле экспоненты равно 2 ^ (e - 1) – 1 + n, если сдвигали влево запятую, 2 ^ (e - 1) – 1 – n, если двигали вправо запятую. Поле мантиссы равно соответствующим битам после запятой с учётом округления. Если после бит мантиссы округляемого 1, то мы пишем число больше него, число меньше него, и записываем в мантиссу то число, которое заканчивается на 0 (в последнем бите мантиссы). Если после бит мантиссы округляемого 0, то просто записываем округляемое. Первый же бит 1, если число отрицательное, и 0, если положительное или 0.
7.Диски. Даётся RPM, количество секторов на дорожке N, размер сектора B, размер файла A, время поиска T1, опционально число поверхностей. Тогда время доступа при произвольном размещении (T1 + ½ * 1/RPM * 60000 + 1/RPM * 1/N * 60000) * (A/B) мс, при наилучшем размещении T1 + ½ * 1/RPM * 60000 + 1/RPM * 1/N * 60000 * (A/B) мс. Не забывать про степень двойки в размере файла.
8.Трансляция адресов. Даётся адрес необходимого размера для обращения к каждой из данного количества (n) ячеек. Рассмотрим случай n = 256. Тогда адрес состоит из 8 бит. Даётся размер страницы k, рассмотрим случай k = 16. Тогда vpn состоит из количества битов необходимого для кодировки номера k, то есть 4 бита в нашем случае. Рассмотрим адрес, например, 0xcd. Тогда vpn это c = 1100. Теперь нужно обратиться к состоянию tlb. Для этого мы берём необходимое для кодировки вариантов наборов в tlb (p) количество

младших бит vpn, в нашем случае пусть будет p = 2 набора, тогда последний бит – это номер набора, а остальные три – тег. То есть номер набора 0, тег 6. Мы попали в tlb, если в таком номере набора есть такой тег. В зависимости от валидности там должна присутствовать ppn, на которую мы заменяем vpn, чтобы перейти в кэш. Итого мы получаем в данном случае при ppn = 0. Получается адрес 00001101. Тут в младших битах кодируется количество байтов (s) в строке кэша. В нашем случае s = 4, нужно 2 бита, тогда следующие по старшинству биты адреса кодируют набор, которых в нашем случае l = 8, то есть это 3 бита, и номер набора 011 = 3, и оставшиеся 3 бита — это тег 000 = 0. Если в этом наборе есть такой тег, то мы попали в кэш.

9.Разбор объявлений переменных и функций в двух модулях программы.

Таблица вопросов:

Символ	Входит ли в .symtab	Тип трансляции	Модуль, где определена	Секция, где определена
example	Да/нет (не входят переменные, объявленные внутри функций)	Global/extern/local (Global – объявленные в этом блоке функции и глобальные переменные, extern – функции и глобальные переменные, объявленные в другом блоке и используемые в этом, local – static)	Первый/второй	.text/.bss/.data

10.Два случая: когда в заданной строке есть PC и нет PC.

1) Адрес ссылки — это адрес начала секции + смещение
   Для получения нового значения нужно вычесть адрес ссылки из суммы адреса новой секции и предыдущего значения ссылки
2) Для получения нового значения нужно сложить адрес новой секции и предыдущее значение ссылки

Учесть инвертированный порядок байтов в записи предыдущего значения ссылки

Дополнительно

11.Последовательность вызова служебных программ драйвером компилятор gcc: препроцессор, компилятор, ассемблер, компоновщик.

12.Сильные символы – объявленные глобальные переменные и функции. Слабые символы – необъявленные глобальные переменные.

13.Схемы логических вентилей рисуются через стандартные and, or, xor.

14.Мэйкфайл смотреть на фото в айфоне.

15.При выходе за знаковые границы типа ставится OF, за беззнаковые CF, при получении в знаковом виде отрицательного числа ставится SF.

16.Double выравнивается в ia-32 в Windows по 8 байт, в Linux по 4 байта, long double и там и там по 4.

В x86_64 всё по-другому: Double и long double всегда выравниваются по 8, также как указатели; кроме того, long в linux выравнивается по 8